CoAt-DQN: Convolution and Attention Deep Q-Networks

Nitin Kumar, Seonwoo Kim
University of Colorado Boulder
{Nitin.Kumar-1, Seonwoo.Kim}@colorado.edu

***Abstract*—Post the advent of Deep Q-Learning, which has been successful in solving Reinforcement learning problems with ease and becoming a state of the art technique, it still had issues in how the agent views the world. The underlying model encoding the visual world ie. CNN, has not been able to model the non-local pixels well leading to misses in these relationships.**
We present an architecture CoAt-DQN which combines CNNs with Transformers by combining attention with convolution and stacking of the individual layers in relevant fashion enabling the advantages of generalization and attention leading to network with better approximation of the Q-values for different actions and the given state, leading to better policies with the improvements making the agent view the world better.
We showcase the reward improvements using Atari Breakout game leading to huge performance improvements especially after 10,000 episodes. We compare the reward outputs with our baseline models DQN and Dueling DQN. Our model, CoAt-DQN shows the fastest improvement and achieves the highest score among the models with 60.8 as its best score for Breakout in 10,000 episode training. Further, We also train our CoAt-DQN model with different batch sizes of 32, 128 and 1024 to further optimize the model. We observe similar performances w.r.t. rewards but training time decreases with increase in batch size.

I. Introduction

Deep Q-Learning Network (DQN) in [1] has been a great success recently with its results proved by a large number of Atari 2600 games. It approximates the Q-values using Neural Networks. Convolutional Neural Networks (CNNs) are integrated to DQN to model the visual information (ie. state of the environment). It is able to extract the local visual features of the state but it does not model non-local pixels well leading to missing relationships among them. CNNs do have great generalization properties with faster converging speed due to good estimation of the inductive bias.

With the introduction of Transformers in [11], which are based on neural network, the encoding of the visual features has improved due to its features of self attention at multiple layers. It attends to the visual information selectively by focusing on what is important, implemented using the self attention feature. It allows the selection and combination of attention information from multiple layers. It has outperformed RNNs for ex. LSTM in [5] and GRU in [6] to model the sequences.

Although, Transformers have large model capacity, they still lack in generalization compared to CNNs since they lack in approximating the inductive biases. They do not work well in the low data regime and fall behind CNNs. Thus, CNNs bring the generalization capability whereas Transformers bring the model capacity yet they have their own issues of lacking attention and right inductive biases respectively. Thus, We want to combine CNNs and Transformers to bring the best of both the worlds and to solve the complementary issues they posses by combining them. Combining them posses another challenge of managing the tradeoff between the efficiency and accuracy which can be solved by merging the deptwise convolution into attention layers using relative attention and stacking convolutional and attention layers in [8]. Finally, Combining the convolution and transformer attention effectively using coatnet improves the encoding of the images used as input to approximate the mapping of the states with actions in DQN and thus modeling it more effectively enhances the mapping of the visual world to actions. It makes the agent view the environment better, leading to better policies and in effect the selection of better actions.

Thus, We propose a Convolution and Attention Deep Q-Learning Network (CoAt-DQN) which unifies the capability of Convolution Networks and Transformers. It effectively brings the generalization capability of CNNs and model capacity of Transformers together to enhance the modeling of the visual information critical for approximation of mapping of actions with states in Deep Q-learning Network. We introduce CoAtNet (Convolution and Attention Network) in [8] to DQN architecture (specifically DQN as well as Duelling DQN in [4]) by replacing the convolution layer with CoAtNet. We show that our model CoAt-DQN outperforms DQN and Dueling-DQN by a huge margin. With 10000 episodes of training, It reaches a reward 60.8 surpassing Dueling-DQN by around 11 percent and almost being 3 times to the score of DQN. We also expect it to converge much faster than these baseline models which can be tested in future with further training. We also did hyperparameter tuning to further optimize our CoAt-DQN model. It showed similar performances for different batch sizes but took lesser time to train as we increased the batch size.

II. Related Work

Since the introduction of Deep Neural Networks, several works have proposed applying deep learning to reinforcement learning. For example, Deep Q-Networks in [1] was introduced by adopting Q-learning concepts. It succeeded to outperform human-level baseline performance in Atari games by only taking image information from game screens. It was a groundbreaking discovery in reinforcement learning that it takes only visual information besides any scalar information. Also, it introduces replay memory and target network to deal with existing issues in training. Since then, various approaches inspired by DQN have been proposed. Van Hasselt who previously proposed Double Q-Learning of course applied a similar concept on DQN and proposed Double DQN in [9] to solve the overestimation issue by using two different networks. Also, Dueling DQN in [4] was proposed with the idea of splitting the deep neural networks into value-network and advantage networks. The value network represents the current state value and the advantage network stands for the relative value of action in a certain state which helps reduce noise and improve its performance. They are different in their architecture but commonly use convolutional layers as their image representation.

However, DQNs still pose the problem of long-term dependency in the visual features for the approximation of Q-function. To deal with this, Transformer in [11] which is motivated from time series based models started drawing attentions. Transformer was initially proposed to solve long term memory issues in time-series models. RNN which was dominant in some past period has a chronic problem of vanishing gradient which makes it hard to feed long sequences. There are some approaches to solve this like LSTM in [5] and GRU in [6] by using memory cells in their architectures, but couldn't perfectly solve the issues. RNN with soft attention in [10] was introduced to handle the long term dependency as well. It worked well but still faced the problem of vanishing gradients. Later, Transformers were introduced which worked effectively to model attention in [11]. It used self-attention network and performed even better than RNNs with attention.

The transformer was successful in time-series tasks and have been applied in different tasks as well. Especially, vision tasks also tried to apply transformer in their models. It led to the replacement of CNN with transformer. Vision transformer (ViT) adapted transformers in [12] but its use in RL has been really costly due to its quadratic complexity with respect to image size of the input. Also, It does not perform well for low data regime compared to CNNs. It has also been suggested that Transformer layers may lack desirable inductive biases. Swin Transformer in [3] improved ViT both computationally as well as in accuracy, although it still is not able to generalize well due to the large model size.

Decision Transformer in [7] has been another recent work that adapted transformer and used it to model complete RL problem using causally masked transformer discarding the use of value function as well as policy gradient methods of RL. It has the limitation that it only works for offline RL. Moreover, Its convergence is not guaranteed when the environment is not explored fully.

CoAtNet in [8] combines the convolution and attention bringing the best of both architectures ie. generalization from CNNs and model capacity from Transformers. It also points out that convolution layers tend to generalize better and converge faster due to strong prior inductive bias. Attention layers provide higher model capacity which is advantageous to model large datasets. It merges depthwise convolution into attention layers with relative attention and stacks the convolution and attention layers in order which brings the strengths from both CNNs and Transformers.

III. Methods

In this section, we present CoAt-DQN, which combines convolution and attention on top of Deep Q-Learning Network. From this architecture, the model can be benefited from both convolution networks and transformers. The architecture of the model is summarized in Figure 1 and Algorithm 1. Also, we explain how we implement the experiments, training, and comparison.

CoAt-DQN Overview. Similar to DQN [1], We encode the visual information using the neural network and train the neural network to map the visual input (ie. state of the environment) to the actions which acts as the output of our neural network. The output produced gives the Q-values for the individual actions given the visual information as the state. With a single forward pass, the network estimates the Q-values for each possible actions. We build CoAt-DQN by replacing the CNN architecture with CoAtnet in vanilla DQN. CoAtnet combines the convolution and self attention in one block and then, computational blocks are stacked to form a complete network structure. CoAtnet combines the translational equivalence property of CNNs and input-adaptive weighing as well as Global receptive field from Transformers.

In order to combine the convolution and self attention, a global static convolution kernel is summed up with adaptive attention matrix before or after softmax normalization of the computed values following equation 1 and equation 2. Attention weights $A_{i,j}$ is computed by the combination of w_{i-j} of the translational equivariance and input-adaptive $x_i^T x_j$, thus catering to both the properties depending on their relative values. Also, Applying relative attention to raw input images makes the computation really slow due to large number of pixels in images and the quadratic complexity of global context with respect to spatial size. Thus, A downsampling is done to reduce the spatial size and thereafter global relative attention is introduced. For downsampling, a multi-stage network with gradual pooling (as in ConvNets) is introduced. A network of 5 stages (ie. S0, S1, S2, S3, S4) are constructed with spatial resolution gradually decreasing from S0 to S4. Spatial size is reduced by 2x with every stage. For first two block convolutional layers are used (due to spacial size being too large for global attention) and thereafter transformers (optionally) are introduced wih global attention. In effect, it leads to the following variants ie. C-C-C-C, C-C-C-T, C-C-T-T and C-T-T-T. It caters to the idea that convolutions work well for non-local pixels which are common in early stages.

C-C-C-C has the highest generalization capability whereas C-C-T-T has the highest model capacity. In effect, C-C-T-T covers all the ticks of generalization, model capacity and efficiency and thus is used for the CoAtnet implementation.

$$y_i^{\text{post}} = \sum_{j\in\mathcal{G}} \left(\frac{\exp\left(x_i^\top x_j\right)}{\sum_{k\in\mathcal{G}} \exp\left(x_i^\top x_k\right)} + w_{i-j} \right) x_j \tag{1}$$

$$y_i^{\text{pre}} = \sum_{j\in\mathcal{G}} \frac{\exp\left(x_i^\top x_j + w_{i-j}\right)}{\sum_{k\in\mathcal{G}} \exp\left(x_i^\top x_k + w_{i-k}\right)} x_j \tag{2}$$

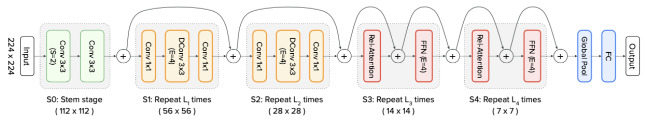

Fig. 1: CoAtnet Architecture [8].

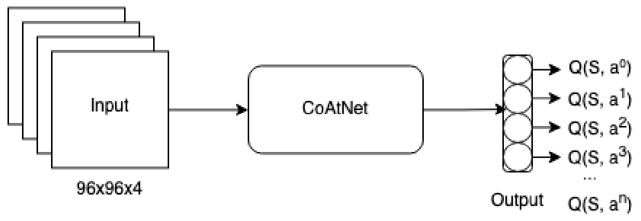

Fig. 2: CoAt-DQN Model Architecture. We modified the CoAtNet network to take 96x96 and 4 channel inputs to utilize replay memory, and its outputs are Q values for the actions for a given state.

IV. Experimental Details

Our experiments are conducted with the Breakout game [14] published by Atari in 1976. We built a Breakout environment for the experiment. Breakout in [14] is one of the Atari games presented as a representative outperforming game in DQN paper [1]. In this game, stacked layers of bricks are aligned on top of the screen and the user can control a board in the horizontal direction to hit a ball. The goal is to destroy all the bricks on the screen. This game is emulated and provided by OpenAI GYM in [13] but requires some modifications to be adapted to our architecture.

For image processing, we render the environment with 'RGB-array' mode and change the RGB to grayscale to reduce the image's dimension. After that, we sliced the top and bottom of the screen which is not directly related to the game information. (The score on the top is already provided by 'reward' in env.step() output). As for the screen size, we change it from 84x84 to 96x96 since CoAt-DQN is optimized with 32x size.

We unwrap the environment and define State-Holder to take 4 past frames as a single state (refer to Figure 2). The State-Holder has initialized with the same initial reset image 4 layers. As the game goes by, new game screen image is queued while the last layer is being popped and removed. With this, we can keep (1, 4, 96, 96) shape of state.

Also we define Replay-Memory as well. This concept was proposed in [1] to overcome correlation issues in game training. It is initialized with an empty list and keeps queuing newly added (state, action, reward, next-state) information. And based on batch size provided, it returns randomly sampled memory later on.

Its reward is presented on top of the screen and also provided by the GYM environment. So we clip the screen information and use the value retrieved from the environment.

The list of chosen parameters for the experiment is shown in Table 1.

Parameter	Value
Frame size	96
Memory size	50,000
Learning rate	1e-4
Optimization step	4
Target Network update every	1,000
Optimizer	Adam
Initial ϵ	1.0
Final ϵ	0.1
Steps	0.1
Batch size	128(default)
Gamma	0.99
Num of Episode	10,000

TABLE I: Parameters used in the experiment

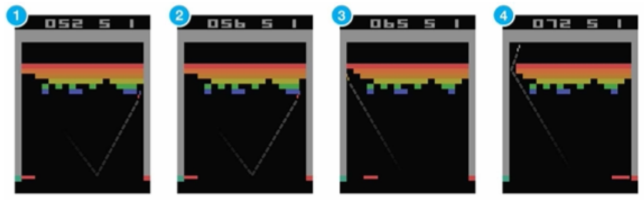

Fig. 3: Visualization of StateHolder to keep past 4 frames in [1]. By utilizing 4 sequential frames, the model can understand the velocity of the ball's movement. In our training, we process RGB to grayscale and stack 4 consecutive frames.

V. Evaluations

Training. The models' agents are tied to the Breakout environment. For each episode, we reset the environment and take action retrieved from the models' output. During the training, the action can be either the models' decision or just a random action based on epsilon values. Each episode is continued until the game is over. Once an episode is done, we update the following based on the pre-set periods; Model optimization, Target network update, and training logs. Lastly, we use 2 RTX3090 GPUs in parallel to speed up the training process. Since the models have image processing blocks(CNN), it takes a too long time to train with a single GPU. We have also used Colab pro with T4 GPU to train for different batch sizes and do hyperparameter tuning.

Baseline model: DQN. We utilize DQN as a baseline for measuring CoAt-DQN's performance. We set DQN model with 3 convolutional layers and 2 fully-connected layers which is widely used for Deep Convolutional Networks. Table 2 contains the detail of DQN model architecture.

Baseline model: DuelingDQN. We additionally train DuelingDQN for the further comparison. DuelingDQN is more advanced than Naive DQN which splits networks into Value networks and Advantage networks. It was well-known for reducing the training noise and getting better convergence than Naive DQN. Table 3 contains the detail of DQN model architecture.

Comparison. We compare CoAt-DQN with a baseline model DQN and DuelingDQN since the CoAt-DQN originated from the baselines. For the direct comparison, we train each model with 10,000 episodes and measure their last 10, 100 average rewards. Also we compare the training progress of

Layer	Hyper-parameters
Convolution	input:4, output:32, kernel:8, stride:4, bias:False
ReLU	-
Convolution	input:32, output:64, kernel:4, stride:2, bias:False
ReLU	-
Convolution	input:64, output:64, kernel:3, stride:1, bias:False
ReLU	-
Flatten	-
Linear	input:4096, output:512
ReLU	-
Linear	input:512, output:4

TABLE II: Baseline DQN model architecture

Layer	Hyper-parameters
Convolution	input:4, output:32, kernel:8, stride:4, bias:False
ReLU	-
Convolution	input:32, output:64, kernel:4, stride:2, bias:False
ReLU	-
Convolution	input:64, output:64, kernel:3, stride:1, bias:False
ReLU	-
Convolution	input:64, output:1024, kernel:7, stride:2, bias:False
ReLU	-
Flatten	-
Linear(Value)	input:512, output:1
Linear(Advantage)	input:512, output:4

TABLE III: Baseline DuelingDQN model architecture

Block type	No. of Blocks	No. of Channels
C	2	64
C	2	96
C	2	192
T	5	184
T	2	768

TABLE IV: CoAt-DQN model parameters. C-MBConv, T-Transformer

each model by plotting the reward histories. We observe that the model's speed of convergence can be affected by its batch size. So we train CoAt-DQN with 3 different batch-size for 5,000 episodes for each. And measure their training progress and the last average rewards.

VI. Results

Comparison with other two baseline models We train the models for 10,000 episodes and log the reward history which is shown in Figure 4 and Table 5. 10000 episodes are not enough to achieve as much performance as what the benchmark shows in [1]. But it is still enough to compare the initial optimization speed. As for the baseline DQN, its performance doesn't even increase until around 6,000 episodes while the other models start being improved earlier than that. CoAt-DQN was the first of the three models to start converging, and Dueling-DQN follows next. At the end of 10,000 episodes, quite significant differences are observed in the training of the three models. CoAt-DQN shows the best performance among the three models with around 50 points, followed by Dueling-DQN.

When we analyze the reward graph closely, it can be observed that CoAt-DQN performs closely in the beginning compared to its baseline models. It can be attributed to the number of parameters that needs to be learnt which is a lot more with the inclusion of transformers and thus it takes a while to gradually learn those parameters and thus performs similar to other baseline models. As the training progresses, the attention weights as well as CNN and Transformer layer weights start to optimize and the gap between the reward performance of the models keep widening. We have only been able to reach 10000 episodes for the training, but as the attention weights are optimized and transformer layers too started to perform well, we expect that at later part of training, the difference between the performance will be even more, with CoAt-DQN surpassing these baseline models with even more margin. Looking at the performance so far, we also expect that the CoAt-DQN model will converge much faster than other models when more training would be done in future.

As shown in Table 5, we analyze the training with focusing on three aspects; when does the model break the 5-average point for the first time in the episode? What is the best grade during the training? The last core. And CoAt-DQN achieves first place for all three aspects. Dueling DQN also achieves similar performance as CoAt-DQN gets. But DQN way falls behind.

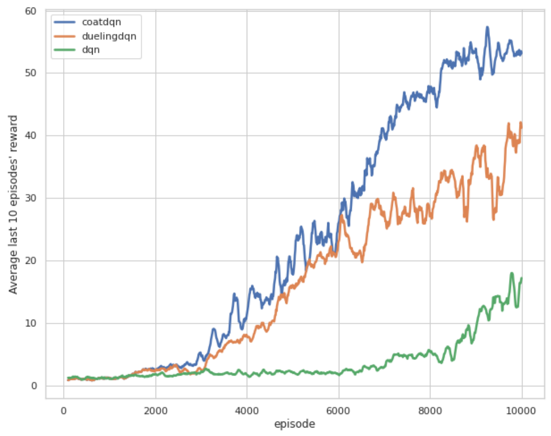

Fig. 4: CoAt-DQN performance compared to the baseline models. The average reward was calculated every 10 episodes and the plot is smoothed with a rolling mean function(10). CoAt-DQN and DuelingDQN start improving faster than DQN. Also for the 10,000 episodes, CoAt-DQN shows out-performing performance compared to the baseline models.

Hyperparameter tuning: Batch size We train the CoAt-DQN model for batch sizes 32, 128 and 1024 to tune the batch size parameter for the model. We observed comparable performance (Figure 5) for the given batch sizes with model of 32 batch size performing better in the beginning whereas model of 1024 batch size getting better later at around 3500 episodes. With increase in batch size the reward curve becomes slightly smoothed inline with our hypothesis although it is not

Models	5 Avg point achieved at	Best score	Last score
CoAt-DQN	**2900**	**60.8**	**52.3**
DuelingDQN	3110	54.1	48.2
DQN	7040	23.0	17.4

TABLE V: CoAt-DQN performance compared to the baseline models. For the three measurements, CoAt-DQN shows the best score in all aspects. Which is close to the DuelingDQN's one and way better than DQN's.

very distinguishable as more training was needed to clearly observe these differences since CoAtDQN model starts to show major performance advantages only after 6000 episodes as can be observed in Figure 4.

Note: Due to Colab pro timeouts at 12h (with GPU), we have only been able to train until then which has limited our results to around 4000 episodes.

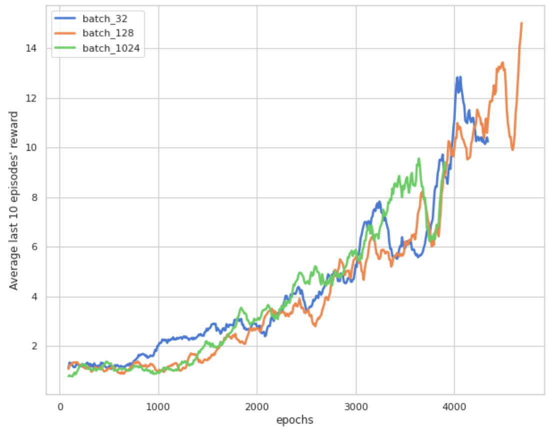

Fig. 5: CoAt-DQN performance comparison for different batch sizes. The average reward was calculated every 10 episodes and the plot is smoothed with a rolling mean function(10). CaAt-DQN performs quite similarly for different batch size except that the time for training decreases with increase batch size.

VII. Discussion

It is evident that CoAt-DQN outperforms all of our baseline models by huge margine attaining to right inductive biases of CNNs, attention and model capacity provided by Transformers. Yet, In order to use it efficiently for Reinforcement learning problems, we need to relook at the computational time and capacity of Transformers. Currently, It takes a lot of time to train CoAt-DQN model (discussed further below) compared to the given baselines due to global attention given to the down-sampled encoded pixels. We would need to further optimize the attention mechanism so that it reduces the training time further, so that it can be used in practical for training. Moreover, Generalization capabilities brought about by the transformers in CoAt-DQN can also be evaluated in future by computing the gap between training loss and the evaluation accuracy. With two models, having the same training loss, the one having better evaluation accuracy will have better generalization capability and it could be a good measure to dig-deeper into the CoAt-DQN's generalization property.

Also, Hyperparameter tuning is required to tune the number of stacked layers of CNNs and Transformer to keep a balance between the accuracy and the training time complexity. As increasing the number of transformer layers increases the time for training with quadratic complexity but also brings its model capacity to optimize the visual encoded information better.

we also faced some issues while training the models which are discussed below:

Training cost. We observe that the training of CoAt-DQN takes much longer than the naive DQN. The increased performance affects the duration for sure but the optimization also takes relatively longer in our experiments. It's because both convolution layer and transformer layer are highly computation-demanding. So it might be helpful to monitor each layer's outputs and adjust the size of the models. The more reducing its size, the more we can get the advantage of computation.

CNN's alternatives. As we discuss above, CoAt-DQN requires a lot of computation including convolution layers. It might be improved by using pre-trained vision networks such as VGG, ResNet, etc. By freezing those layers, or fine-tuning them with memory states online, we can decrease trainable parameters. That leads to optimization process focused on transformer layers. We can also combine pre-trained transformer layers in similar fashion to re-use the benefits of previous training and to reduce the training time for the CoAt-DQN model.

Comparison with policy gradient models. During the research, we've found that there are state-of-the-art policy gradient-based models such as A3C. Some of them were proven to work well in a Breakout environment. Since there are many policy gradient based methods proposed, it'd be fair to compare with them as well. It will give a more holistic view of the improvements observed for our model.

VIII. Conclusion

In this paper, We have combined the properties of CNNs and Transformer to bring the best of both the worlds. Huge performance improvements compared to the baseline models prove the advantage that CoAt-DQN brings in to understand the visual world better. It also showcases the improved approximations of the Q-values for the different actions computed by the model leading to better rewards and faster learning. the right inductive biases of CNNs and attention and model capacity of transformers, in combination indeed seem to have brought these improvements.

With further training, The impact of model capacity of transformers will become even more observable, with the expectation that it can compete quite well with the latest models. It would also mean that, with proper hyperparameter tuning, it can even converge faster than other algorithms. Also, We have only trained and tested our model with Breakout environment, but we can further extend it to other use cases especially where state space is huge and thus the model capacity and attention features of transformers would even be more useful for those scenarios.

References

[1] Mnih, Volodymyr, et al. ”Human-level control through deep reinforcement learning.” nature 518.7540 (2015): 529-533.

[2] Van Hasselt, Hado, Arthur Guez, and David Silver. ”Deep reinforcement learning with double q-learning.” Proceedings of the AAAI conference on artificial intelligence. Vol. 30. No. 1. 2016.

[3] Meng, Li, et al. ”Deep Reinforcement Learning with Swin Transformer.” arXiv preprint arXiv:2206.15269 (2022).

[4] Wang, Ziyu, et al. ”Dueling network architectures for deep reinforcement learning.” International conference on machine learning. PMLR, 2016.

[5] Hochreiter, Sepp, and Jürgen Schmidhuber. ”Long short-term memory.” Neural Computation 9.8 (1997): 1735-1780.

[6] Chung, Junyoung, et al. ”Empirical evaluation of gated recurrent neural networks on sequence modeling.” arXiv preprint arXiv:1412.3555 (2014).

[7] Chen, Lili, et al. ”Decision transformer: Reinforcement learning via sequence modeling.” Advances in neural information processing systems 34 (2021): 15084-15097.

[8] Dai, Zihang, et al. ”Coatnet: Marrying convolution and attention for all data sizes.” Advances in Neural Information Processing Systems 34 (2021): 3965-3977.

[9] H. van Hasselt, A. Guez, D. Silver, ‘Deep Reinforcement Learning with Double Q-learning’, CoRR, abs/1509.06461, 2015.

[10] D. Bahdanau, K. Cho, Y. Bengio, ‘Neural Machine Translation by Jointly Learning to Align and Translate. arXiv, 2014.

[11] A. Vaswani., ‘Attention Is All You Need’. arXiv, 2017.

[12] A. Dosovitskiy., ‘An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale’. arXiv, 2020.

[13] openai, “openai/gym: A toolkit for developing and comparing reinforcement learning algorithms.,” GitHub, Oct. 25, 2022. https://github.com/openai/gym (accessed Dec. 10, 2022).

[14] Wikipedia Contributors, “Breakout (video game),” Wikipedia, Nov. 27, 2022. https://en.wikipedia.org/wiki/Breakout_(video_game) (accessed Dec. 10, 2022).